UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 83-7	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A NOTE ON A COMBINED APPROACH TO THE PALLET LOADING PROBLEM		5. TYPE OF REPORT & PERIOD COVERED Technical 6. PERFORMING ORG. REPORT NUMBER 83-7
7. AUTHOR(s) Thom J. Hodgson Diana Swift Hughes Louis A. Martin-Vega		8. CONTRACT OR GRANT NUMBER(s) (Air Force) F01600-80-D0299 (Navy) N00014-76-C-0096
9. PERFORMING ORGANIZATION NAME AND ADDRESS Industrial and Systems Engineering University of Florida Gainesville, FL 32611		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research - Arlington, VA A. F. Logistics Mgmt. Center Gunter AFS, Alabama		12. REPORT DATE February, 1983 13. NUMBER OF PAGES 17
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Pallet Loading
Stock Cutting
Dynamic Programming
Packing

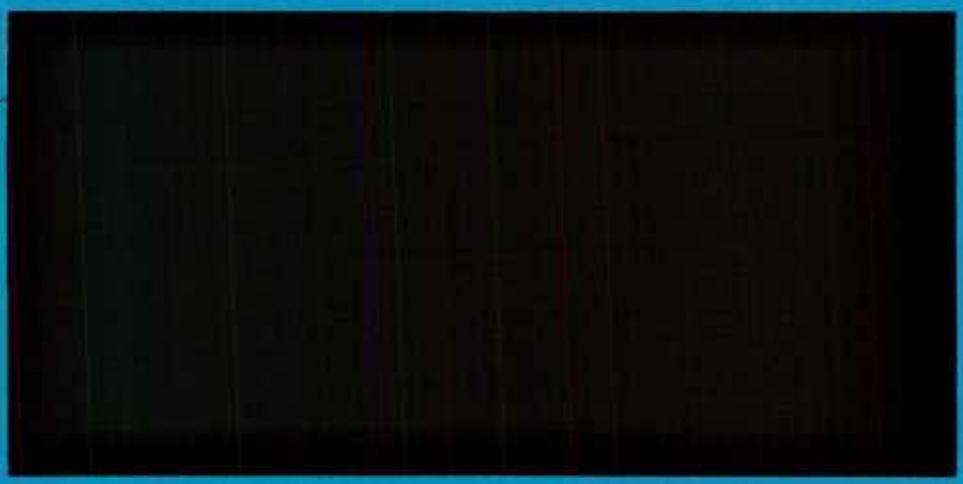
20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

In a recent paper Hodgson developed a dynamic programming based heuristic for the two-dimensional pallet loading problem. This note presents improvements to that procedure which have resulted in reductions of CPU run times of up to 20 to 1 as well as drastic lowering of memory requirements. Other implications of the improvements are also discussed.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)



RESEARCH REPORT

Industrial & Systems
Engineering Department
University of Florida
Gainesville, FL. 32611

A NOTE ON A COMBINED APPROACH TO THE PALLET LOADING PROBLEM

Research Report No. 83-7

by

Thom J. Hodgson
Diana Swift Hughes
Louis A. Martin-Vega

February, 1983

Department of Industrial and Systems Engineering
University of Florida
Gainesville, Florida 32611

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

This research was supported in part by the U.S. Air Force, under contract number F01600-80-D0299, and by the Office of Naval Research, under contract number N00014-76-C-0096.

TABLE OF CONTENTS

A Note on a Combined Approach
to the Pallet Loading Problem

Abstract

In a recent paper Hodgson developed a dynamic programming based heuristic for the two-dimensional pallet loading problem. This note presents improvements to that procedure which have resulted in reductions of CPU run times of up to 20 to 1 as well as drastic lowering of memory requirements. Other implications of the improvements are also discussed.

Introduction

In a recent paper, Hodgson [1] developed a dynamic programming based heuristic for the two-dimensional pallet loading problem. In the course of evaluating the computational results for the procedure, it was noted that the quality of solutions obtained by the procedure (percent coverage of the pallet area) was very good. However, the CPU seconds required to compute solutions were larger than would be desireable for application in a real-time environment. It was observed that there were elements of the procedure that might be enhanced in order to improve computation times. The results of experimental computation indicated that computation time was extremely sensitive to the quality of a bounding function that was used in the procedure. It was conjectured that it might be possible to obtain significant computational improvements if the quality of the bounding function was improved even slightly.

In the following, the dynamic programming based pallet loading procedure is reviewed along with the development of the bounding function. The results of the experimentation with heuristic knapsack procedures is then discussed. A fundamental improvement in the bounding function is developed and another improvement taken from observations of real-world pallet loaders and furniture movers is presented. The computational experiments from the previous paper [1] are duplicated and the results compared.

The Pallet Loading Procedure Reviewed

The reader should note that in the interests of brevity, details of the procedure are not given. The interested reader is referred to the original paper [1]. The key element of the dynamic programming based procedure is that the pallet can be partitioned by a rectangular section (see Figure 1). If a good load can be found for the left-hand sub-pallet (x,y) (upper left-hand rectangle of Figure 1), then a good load for the left-hand sub-pallet (x',y') might be found by packing boxes into the shaded portion of Figure 1 (i.e., sub-pallet (x',y') minus sub-pallet (x,y)). In this way it would be possible to build up to the entire pallet (L,W). In order to consider all possible ways of building up to loading the entire pallet the concepts of dynamic programming are used. The following definitions are useful.

N = Set of all boxes to be considered for loading (of size n).

I = Subset of the boxes, 1,2,...,n.

J = Subset of the boxes, 1,2,...,n.

x,y = Two dimensional index specifying a rectangular partition (Figure 1).

f(x,y,I) = The maximum area of the left-hand subpallet of x,y which can be covered using the subset of boxes I.

h(x,y,x',y',I) = The maximum area of the left-hand sub-pallet of x',y' less the left-hand sub-pallet of x,y (shaded area of Figure 2) which can be covered using boxes from the set I.

The dynamic programming equation for the pallet loading problem can be given as follows:

$$f(x',y',I) = \max_{\substack{x \leq x' \\ x \leq y' \\ J \subseteq I}} \{f(x,y,J)+h(x,y,x',y',I-J)\} \qquad (1)$$

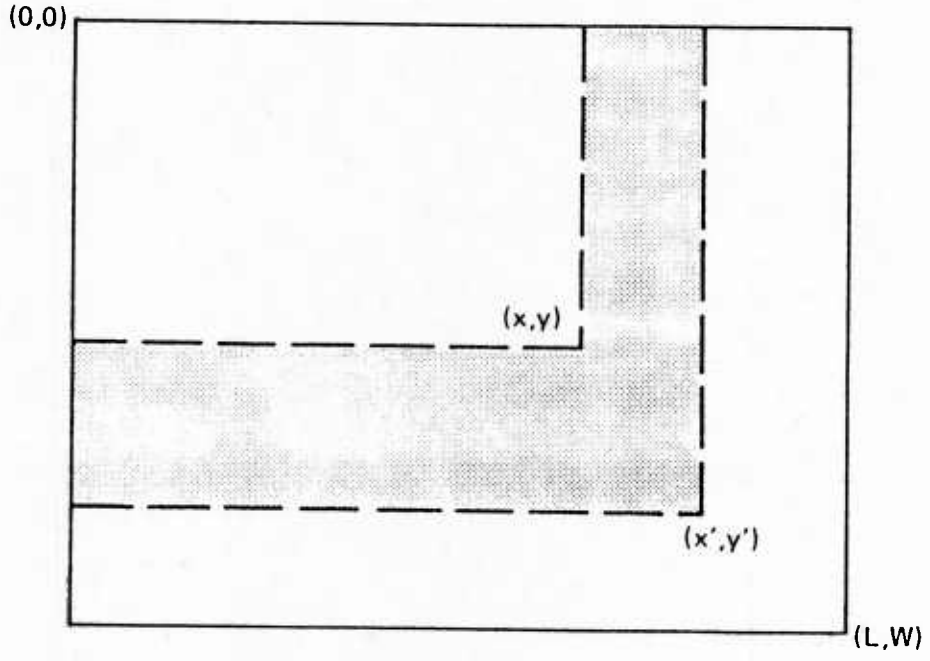


Figure 1: Pallet with Two Rectangular Partitions (x,y) and (x',y').

In order to limit the size of the state space only one partial solution f(x,y,I) for each partition x,y is saved. That is,

$$f'(x,y) = \max_{I}\{f(x,y,I)\}.$$

The function h(x,y,x',y',I) itself requires an optimization in order to pack the L-shaped area (shaded area of Figure 1). This is done simply by breaking the L-shaped partition into rectangular areas and filling each area using a linear dynamic programming knapsack procedure.

In order to eliminate unnecessary computation in the procedure a bounding function is used:

B(L,W) = the maximum possible coverage of a pallet (sub-pallet) of length L and width W using the set of boxes N.

The bounding function is developed by solving relaxations of the pallet loading problem. In the present case, two different relaxations are used. The first is

$$C_n(x) = \text{maximize} \sum_{i=1}^{n} \ell_i x_i + w_i x_{n+i}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} \ell_i x_i + w_i x_{n+i} \leq X$$

$$x_i \cdot x_{n+i} = 0 \qquad i = 1,2,\ldots,n$$

$$x_i = 0,1 \qquad i = 1,2,\ldots,2n$$

Note that ℓ_i = length of box i, and

w_i = width of box i.

The function $C_n(X)$ specifies the maximum linear coverage that is possible on the line segment [0,X] choosing from the set of boxes 1,...,n (positioning them by either length or width). For a pallet (or rectangular sub pallet) of size L by W, an upper bound on the maximum area coverage possible is $C_n(L)*C_n(W)$. The funcion $C_n(X)$ is computed in a straight forward fashion using a dynamic programming knapsack routine.

The second relaxation is

$$D_n(Z) = \text{maximize} \sum_{i=1}^{n} \ell_i \cdot w_i \cdot x_i$$

$$\text{s.t.} \quad \sum_{i=1}^{n} \ell_i \cdot w_i \cdot x_i \leq Z$$

$$x_i = 0,1, \qquad i = 1,2,\ldots,n$$

The function $D_n(Z)$ specifies the maximum area coverage that is possible on a pallet of area Z, choosing from the set of boxes 1,...,n, and assuming that the boxes can be "mashed" into any shape while maintaining constant area. For a pallet (or rectangular sub-pallet) of size L by W, an upper bound on the maximum area coverage possible is $D_n(L*W)$. The function $D_n(L)$, again, is computed in a straightforward fashion using a dynamic programming knapsack routine. The bounding function used by the procedure is

$$B(L,W) = \min\{C_n(L)*C_n(W), D_n(L*W)\}. \qquad (2)$$

Computational Improvements

The first attempt at computational improvement was to incorporate a stronger bounding function. The bounding function can be improved by making the following simple observation:

$$B(L,W) = D_n(C_n(L)*C_n(W)) \leq \min\{C_n(L)*C_n(W), D_n(L*W)\}. \quad (3)$$

Substituting equation (3) for equation (2) results in a substantial reduction in computational effort. Limited experimentation resulted in reductions of up to 4 to 1 in CPU time.

A second attempt at computational improvements came from our observations of people loading pallets in the real world. A typical approach in loading a pallet is to take the largest box that has to be loaded and place it in one corner. Then the pallet load is built around the corner box. In the previous procedure every box in the set is a candidate to be placed in the corner, thereby greatly increasing the combinatorics involved in loading the pallet. This was noted previously [1] and the original version of the procedure allowed the user to dictate the corner box, if so desired. In the present case, the procedure was modified so that, unless the user desires otherwise, the box with the largest area coverage is always placed in the corner.

Experimental Computation

The improved pallet loading procedure was programmed in Fortran IV and implemented on the University of Florida IBM 3033. Problems with a data set of 30 boxes were generated with box dimensions uniformly distributed (integer values only) between upper and lower bounds as indicated in Tables 1, 2, and 3. Table 1 contains computation times in virtual seconds. Table 2 contains the number of undominated partial solutions generated by the dynamic program. Table 3 contains the percent area of the pallet covered. Each entry of the tables is the average of five problems. The experimentation duplicates that of the previous paper [1]. The upper left-hand entries of each box are from the previous experimentation. The lower right-hand entries are from the present experimentation.

As with the previous experimentation the computation times are clearly dependent on both pallet size and the range of box dimensions (Table 1). What is interesting is the overall reduction in computational effort. The relative reduction in computation increases with both the size of the problem and the difficulty (range of box sizes) with a maximum reduction of almost 20 to 1. It should be noted that it is not possible to compare times for the most difficult problem sets since the original procedure was not capable of solving the problems in 30 seconds of computation.

The number of undominated partial solutions (a measure of the storage requirements of the procedure) is given in Table 2. The reductions obtained are due solely to placing only one candidate box in the corner to start the procedure as the improved bounding function does not affect the number of undominated solutions. While the growth in requirement for computer storage is also dependent on pallet size and the range of box dimensions, reductions ranging from the order of 10 to 1 or better for the smaller pallet sizes and 3

TABLE 1: COMPUTATION TIMES IN VIRTUAL SECONDS

Box Size	AREA PALLET DIMENSIONS				
Lower Bounds / Upper	2400 60 x 40	3500 70 x 50	4800 80 x 60	6400 80 x 80	8000 100 x 80
18 / 22	.039 / .0156	.155 / .113	.674 / .256	1.60 / .686	6.07 / 2.31
17 / 23	.081 / .0244	.350 / .147	1.60 / .401	3.48 / .867	13.537 / 4.05
16 / 24	.140 / .0337	.864 / .216	2.73 / .588	6.33 / 1.340	25.44 / 5.80
15 / 25	.193 / .0604	1.437 / .294	5.13 / .866	9.07 / 1.69	* / 8.583
10 / 30	3.112 / .3319	20.136 / 1.017	* / 3.79	* / 7.12	* / 28.482

*Not solved in 30 virtual seconds.

TABLE 2: UNDOMINATED PARTIAL SOLUTIONS GENERATED

Box Size	AREA PALLET DIMENSIONS				
Lower Bounds Upper	2400 60 x 40	3500 70 x 50	4800 80 x 60	6400 80 x 80	8000 100 x 80
18 / 22	57.0 / 9.2	65.2 / 12.6	207.4 / 57.2	263.4 / 84.8	554.8 / 222.6
17 / 23	97.0 / 11.6	126.2 / 18.0	384.6 / 93.2	484.0 / 146.0	1059.0 / 398.2
16 / 24	135.4 / 15.8	238.8 / 26.8	591.4 / 147.8	769.4 / 238.8	1551.5 / 611.0
15 / 25	174.8 / 19.6	379.6 / 35.6	766.6 / 200.2	988.6 / 294.0	* / 774.0
10 / 30	423.0 / 27.6	842.2 / 107.0	* / 443.0	* / 603.0	* / 1281.0

* Not solved in virtual seconds.

TABLE 3: PERCENT COVERAGE OF THE PALLET AREA

Box Size	AREA PALLET DIMENSIONS				
Lower Bounds / Upper	2400 60 x 40	3500 70 x 50	4800 80 x 60	6400 80 x 80	8000 100 x 80
18 / 22	100 / 100	75.5 / 78.9	100 / 100	100 / 100	99.95 / 100
17 / 23	100 / 99.6	83.5 / 86.1	100 / 100	99.94 / 99.9	100 / 99.8
16 / 24	100 / 99.6	93.2 / 92.75	100 / 100	99.95 / 99.9	99.93 / 99.76
15 / 25	100 / 99.0	99.0 / 98.1	100 / 99.85	99.84 / 99.81	* / 99.3
10 / 30	99.5 / 97.7	99.7 / 99.1	* / 99.5	* / 99.23	* / 99.33

*Not solved in 30 virtual seconds.

to 1 for the larger pallet sizes have been obtained thereby significantly enhancing the potential applicability of the procedure on computers of limited storage capacity.

The quality of the solutions for the randomly generated problems is measured by the percent of the pallet area covered (Table 3). Four out of the five pallet sizes used in the experimentation (60 x 40, 80 x 60, 80 x 80, and 100 x 80) were chosen so that there would be a high likelihood of the existance of an extremely good solution. Across all comparable solutions there was an average loss of 0.26% in terms of area coverage. This is caused by the limitation imposed by placing the largest box in the corner. The effect is greatest in the smallest pallet size (60 x 40), as might be expected. Considering only the larger pallets (80 x 60 and larger), the loss in area coverage is 0.054% on the average. In our interactive experiments, where the user was allowed to choose the corner box, the loss in area coverage is virtually zero.

One pallet size used in the experimentation (70 x 50) was chosen so that there would not be a high likelihood of good solutions. With all boxes having dimensions of 20 x 20, the best possible coveage would be 68.6%. With box dimensions randomly distributed from 18-22, the best possible coverage (from an infinite set of boxes) would be 83.0% (versus 78.9% attained). With box dimensions randomly distributed from 17-23, the upper bound would be 92.0% (versus 86.1% attained), and for all other problems the upper bound would be 100%. It might be noted that for the 18-22, and 17-23 problems (and the 18-22 problem on a 100 x 80 pallet), there is actually an increase in the percent coverage. This is because the original version of the knapsack subproblem considered only 15 boxes at a time. The current version was expanded to consider up to 30 boxes at a time.

In synopsis, it is important to point out that the results presented in Table 1 have been based on box sizes with uniformly distributed dimensions. Real loading problems seldom have uniformly distributed box dimensions, but rather tend to have lengths, widths and heights that cluster at particular values. Generally, for real problems, the bounding function, generates even stronger bounds than those obtained in our experimentation resulting in even greater reductions in computation times. The results in Table 2 demonstrate the effect of limiting the choice of the candidate corner box and how this has resulted in reduced storage requirements. Finally, Table 3 shows that the price paid in terms of the quality of the solutions obtained is relatively insignificant, allowing one to conclude that the improvements are in fact a significant step forward in terms of the use of the procedure in a real-time environment.

Reference

[1] Hodgson, Thom J., "A Combined Approach to the Pallet Loading Problem", IIE Transactions, Vol. 14, No. 3, Sept., pg. 175 (1982).